What is claimed is:

[Claim 1] 1. In a database system, a method for parallel optimization of a query, the method comprising:

generating a plurality of parallel plans for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel;

adjusting parallel operators of each parallel plan if necessary based on resources available for executing the query;

creating a schedule for each parallel plan indicating a sequence for execution of operators of each parallel plan;

determining execution cost of each parallel plan based on its schedule; and

selecting a particular parallel plan having lowest execution cost for obtaining data requested by the query.

[Claim 2] 2. The method of claim 1, wherein the query comprises a Structured Query Language (SQL) expression.

[Claim 3] 3. The method of claim 1, wherein said generating step includes generating an operator tree for each parallel plan based on the query.

[Claim 4] 4. The method of claim 3, wherein said step of generating an operator tree includes generating nodes of the operator tree as iterators for applying predefined behavior to data.

[Claim 5] 5. The method of claim 3, wherein said step of generating an operator tree includes inserting a parallel operator in the operator tree.

[Claim 6] 6. The method of claim 5, wherein said step of generating an operator tree includes dividing a query operation into sub-tasks and said parallel operator provides for executing said sub-tasks in parallel.

[Claim 7] 7. The method of claim 6, wherein said step of executing said sub-tasks in parallel includes executing said sub-tasks in parallel across a plurality of storage units.

[Claim 8] 8. The method of claim 6, wherein said step of executing said sub-tasks in parallel includes executing said sub-tasks in parallel across a plurality of CPUs.

[Claim 9] 9. The method of claim 1, wherein said parallel operator provides for pipelining of intermediate results from a first set of operators to a second set of operators.

[Claim 10] 10. The method of claim 1, wherein said generating step includes generating a parallel plan using a partitioning property so as to efficiently partition data among operators of the parallel plan.

[Claim 11] 11. The method of claim 1, wherein said generating step includes generating a cost vector for each parallel plan.

[Claim 12] 12. The method of claim 11, wherein said cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator in the parallel plan, and resource usage of an operator in the parallel plan for a certain time period.

[Claim 13] 13. The method of claim 11, wherein said generating step further comprises:

pruning a first parallel plan having a cost vector less favorable in each vector dimension than a second parallel plan.

[Claim 14] 14. The method of claim 1, wherein said generating step includes generating a plurality of parallel plans based at least in part on partitioning and multi-dimensional costing.

[Claim 15] 15. The method of claim 1, wherein said adjusting step includes adjusting a parallel plan for available worker processes at compile time.

[Claim 16] 16. The method of claim 15, wherein the parallel plan comprises an operator tree and an adjustment is made to at least some parallel operators of the operator tree if the operator tree exceeds maximum configured worker processes.

[Claim 17] 17. The method of claim 1, wherein said step of adjusting parallel operators of each parallel plan if necessary based on resources

available for executing the query includes adjusting parallel operators based on available memory resources.

[Claim 18] 18. The method of claim 1, wherein said creating step includes separating a resource intensive operator into a plurality of operators.

[Claim 19] 19. The method of claim 1, wherein said creating step includes identifying pipelines in each parallel plan.

[Claim 20] 20. The method of claim 19, wherein said creating step includes constructing a pipeline dependency tree based on dependencies among operators of each parallel plan.

[Claim 21] 21. The method of claim 20, wherein said creating step includes determining order of execution of pipelines based on the pipeline dependency tree and available resources.

[Claim 22] 22. The method of claim 19, further comprising:

if resource usage of a particular pipeline is greater than resources available for the particular pipeline, splitting the particular pipeline into a plurality of pipelines.

[Claim 23] 23. The method of claim 22, wherein said step of splitting the particular pipeline includes adding operators for materializing the particular pipeline into a plurality of pipelines at intervals such that resource usage is evenly distributed over the plurality of pipelines.

[Claim 24] 24. A computer-readable medium having processor-executable instructions for performing the method of claim 1.

[Claim 25] 25. A downloadable set of processor-executable instructions for performing the method of claim 1.

[Claim 26] 26. A system for parallel optimization of a database query, the system comprising:

a search engine for generating a plurality of parallel plans which can be used for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel;

a parallel scheduler for adjusting parallel operators of each parallel plan if necessary based on resources available for executing the query and creating a schedule for the parallel plan indicating a sequence for execution of operators of the parallel plan; and

a module for determining execution cost of each parallel plan based on its schedule and selecting a particular parallel plan having lowest execution cost for obtaining data requested by the query.

[Claim 27] 27. The system of claim 26, wherein the query comprises a Structured Query Language (SQL) expression.

[Claim 28] 28. The system of claim 26, wherein the search engine generates an operator tree for each parallel plan based on the query.

[Claim 29] 29. The system of claim 28, wherein the search engine generates nodes of the operator tree as iterators for applying predefined behavior to data.

[Claim 30] 30. The system of claim 28, wherein the search engine inserts a parallel operator in the operator tree.

[Claim 31] 31. The system of claim 30, wherein the search engine divides a query operation into sub-tasks and said parallel operator provides for executing said sub-tasks in parallel.

[Claim 32] 32. The system of claim 31, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of storage units.

[Claim 33] 33. The system of claim 31, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of CPUs.

[Claim 34] 34. The system of claim 26, wherein said parallel operator provides for pipelining of intermediate results from a first set of operators to a second set of operators.

[Claim 35] 35. The system of claim 26, wherein the search engine generates a parallel plan using a partitioning property so as to efficiently partition data among operators of the parallel plan.

[Claim 36] 36. The system of claim 26, wherein the search engine generates a cost vector for each parallel plan.

[Claim 37] 37. The system of claim 36, wherein said cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator in the parallel plan, and resource usage of an operator in the parallel plan for a certain time period.

[Claim 38] 38. The system of claim 36, wherein the search engine prunes a first parallel plan having a cost vector less favorable in each vector dimension than a second parallel plan.

[Claim 39] 39. The system of claim 26, wherein the search engine generates a plurality of parallel plans based at least in part on partitioning and multi-dimensional costing.

[Claim 40] 40. The system of claim 26, wherein the parallel scheduler adjusts a parallel plan for available worker processes at compile time.

[Claim 41] 41. The system of claim 26, wherein a parallel plan comprises an operator tree and the parallel scheduler adjusts at least some parallel operators of the operator tree based on available threads.

[Claim 42] 42. The system of claim 26, wherein the parallel scheduler separates a resource intensive operator into a plurality of operators.

[Claim 43] 43. The system of claim 26, wherein the parallel scheduler identifies pipelines in the parallel plan.

[Claim 44] 44. The system of claim 43, wherein the parallel scheduler constructs a pipeline dependency tree based on dependencies among operators of a parallel plan.

[Claim 45] 45. The system of claim 44, wherein the parallel scheduler determines order of execution of pipelines based on the pipeline dependency tree and available resources.

[Claim 46] 46. The system of claim 43, wherein the parallel scheduler splits a particular pipeline into a plurality of pipelines.

[Claim 47] 47. The system of claim 46, wherein said parallel scheduler adds operators for materializing the particular pipeline into a plurality of pipelines at intervals such that resource usage is evenly distributed over the plurality of pipelines.

[Claim 48] 48. A method for parallel optimization of a query requesting data from a database, the method comprising:

creating a plurality of operator trees for executing the query, the operator trees providing for execution of portions of the query in parallel;

adjusting the portions of the query to be executed in parallel based on memory resources available for executing the query;

generating a schedule for execution of each operator tree; and

selecting the operator tree having lowest execution cost based on its schedule for executing the query.

[Claim 49] 49. The method of claim 48, wherein the query comprises a Structured Query Language (SQL) expression.

[Claim 50] 50. The method of claim 48, wherein said creating step includes creating an operator tree including parallel operators for execution of portions of the query in parallel.

[Claim 51] 51. The method of claim 50, wherein said parallel operators comprise iterators for applying predefined behavior to data.

[Claim 52] 52. The method of claim 50, wherein said step of creating an operator tree includes creating operators for tasks to be performed in executing the query and said parallel operator provides for executing said tasks in parallel.

[Claim 53] 53. The method of claim 50, wherein a parallel operator executes in parallel across a plurality of storage units.

[Claim 54] 54. The method of claim 50, wherein a parallel operator executes in parallel across a plurality of CPUs.

[Claim 55] 55. The method of claim 50, wherein a parallel operator provides for pipelining of intermediate results from a first set of operators to a second set of operators.

[Claim 56] 56. The method of claim 48, wherein said creating step includes creating an operator tree using a partitioning property so as to efficiently partition data among operators.

[Claim 57] 57. The method of claim 48, wherein said creating step includes generating a cost vector for each operator tree.

[Claim 58] 58. The method of claim 57, wherein said cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator, and resource usage of an operator for a certain time period.

[Claim 59] 59. The method of claim 57, wherein said creating step further comprises:

pruning a first operator tree having a cost vector less favorable in each vector dimension than a second operator tree.

[Claim 60] 60. The method of claim 48, wherein said creating step includes creating a plurality of operator trees based at least in part on partitioning and multi-dimensional costing.

[Claim 61] 61. The method of claim 48, wherein said adjusting step includes adjusting an operator tree for available worker processes at compile time.

[Claim 62] 62. The method of claim 48, wherein said operator tree includes parallel operators for executing portions of the query in parallel and said adjusting step includes adjusting said parallel operators if necessary based on available memory resources.

[Claim 63] 63. The method of claim 48, wherein said adjusting step includes separating a resource intensive operator into a plurality of operators.

[Claim 64] 64. The method of claim 48, wherein said generating step includes identifying pipelines in each operator tree.

[Claim 65] 65. The method of claim 64, wherein said generating step includes constructing a pipeline dependency tree based on dependencies among operators of each operator tree.

[Claim 66] 66. The method of claim 65, wherein said creating step includes determining order of execution of pipelines based on the pipeline dependency tree and available resources.

[Claim 67] 67. The method of claim 66, further comprising:

if resource usage of a particular pipeline is greater than resources available for the particular pipeline, splitting the particular pipeline into a plurality of pipelines.

[Claim 68] 68. The method of claim 67, wherein said step of splitting the particular pipeline includes adding operators for materializing the particular pipeline into a plurality of pipelines at intervals such that resource usage is evenly distributed over the plurality of pipelines.

[Claim 69] 69. A computer-readable medium having processor-executable instructions for performing the method of claim 48.

[Claim 70] 70. A downloadable set of processor-executable instructions for performing the method of claim 48.